УДК 378:004.384

ПОРІВНЯННЯ ХАРАКТЕРИСТИК МОВ ПРОГРАМУВАННЯ ЗА ДОПОМОГОЮ АЛГОРИТМІЧНИХ РОЗРАХУНКІВ.

М.Г. Бердник, О.А. Сподинець, Д.М. Барвіненко
(Україна, Дніпро, НТУ «Дніпровська політехніка»)

У кожної мови програмування є свої переваги і недоліки. Одна з найважливіших характеристик транслятора з будь-якої мови - це швидкість виконання програм. Дуже важко або навіть неможливо отримати точну оцінку такій швидкості виконання. Граничну ємність стека, критичну величину для

рекурсивних обчислень перевірити простіше, але вона може змінюватися в різних версіях транслятора і бути залежною від системних налаштувань.

Тестувалися такі транслятори: c (gcc, clang, icc), javascript (Google Chrome, Mozilla Firefox), Lisp (sbcl, clisp), Erlang, Haskell (ghc, hugs), Lua, Ruby, PHP, Bash. Досліджувалися як власне швидкість виконання декількох невеликих, але трудомістких алгоритмів, так і:

- якість оптимізації трансляторів;
- особливості при роботі з процесорами Intel і AMD;
- граничне число рекурсивних викликів (ємність стека).

У якості першої задачі, на якій тестувалися всі транслятори, обраний розрахунок числа Фібоначчі подвійний рекурсією згідно з визначенням: числа з номерами 1 і 2 - це одиниці, а наступні - це сума двох попередніх. Цей алгоритм має кілька привабливих особливостей:

- Якщо час розрахунку n-го числа t, то (n + 1) -го - t * φ, де φ - це золотий перетин рівний (√5 + 1) / 2;
- Саме обчислюється n-е число рівне округленому до найближчого цілого величиною φn /√5;
- Розрахунок fib (n + 1) вимагає n-й вкладеності викликів.

Перша особливість дозволяє за невеликий час протестувати транслятори, швидкості роботи яких розрізняються в сотні тисяч разів. Друга особливість дозволяє швидко перевіряти правильність розрахунків. Третя особливість теоретично дозволяє досліджувати ємність стека, але через те, що розрахунок при n> 50 стає дуже повільним навіть на суперкомп'ютері, практично використовувати цю особливість не представляється можливим.

В Таблиці 1 відображені результати. В другій колонці вказується назва мови, назва компілятора і його версія і, якщо використовувалася, опція оптимізації генерованого коду. У третій колонці наводиться відносне час обчислення на процесорі AMD Phenom II x4 3.2 ГГц. Тести проводилися і на AMD FX-6100 на такій же частоті, але їх результати мало відрізняються від наведених. В 4-й колонці наводиться відносне час обчислення на процесорі Intel Core i3-2100 3.1 ГГц. Так як порівняння процесорів не було метою дослідження, частина трансляторів були протестовані на платформі Intel. У п'ятій - оцінка зверху (точність 10%) максимального числа рекурсивних викликів, які підтримуються транслятором при обчисленні fib (1,1, n) на комп'ютері з 8 Гб оперативної пам'яті з розміром системного стека (ulimit -s) 8192 КБ. Деякі транслятори використовують власні настройки, які визначають розмір використовуваного стека - завжди використовуються значення за замовчуванням для обраної версії транслятора. Виміри проводилися в системі Linux, але їх результати не повинні змінюватися при переході до іншого ОС. Дані відсортовані за 3-й колонкою.

Таблиця 1

N	Мова	AMD	Intel	Стек
1	C/C++ (gcc 4.7.2, -O5)	354056	493533	790000
2	C/C++ (clang 3.0-6.2, -O3)	307294		270000
3	C/C++ (icc 14.0.3, -fast)	250563	232665	53000
11	Javascript (Mozilla Firefox 25)	121979		4200
12	Javascript (Google Chrome 31)	92850		10000
13	Lisp (sbcl 1.0.57)	54925	51956	31000
14	Erlang (5.9.1)	19845	18589	межі немає
15	Haskell (ghc 7.4.1, -O)	18589	22946	260000
17	Lua (5.2)	6420	7075	150000
18	Ruby (1.9.3)	5297	6969	6600
22	PHP (5.4.4)	2822	3720	межі немає
28	Lisp (clisp 2.49)	998	1023	5500
33	Haskell (hugs 98.200609.21)	82	121	17000
35	bash (4.2.37)	1	0,77	600

В якості другої задачі обрана функція Аккермана в формі, коли до неї зводяться всі арифметичні операції, тобто Ack (1, x, y) = x + y, ack (2, x, y) = x * y, ack (3, x, y) = xy, ack (4, x, y).

Ця функція з ростом n росте дуже швидко, але вважається дуже повільно. Остання властивість теоретично зручно для тестування швидкодії. Однак, розрахунок цієї функції вимагає значного числа рекурсивних викликів і більшість тестованих мов виявилося не в змозі їх підтримувати для обчислень, що мають помітну тривалість. Відомо, що обчислення цієї функції не можна звести до ітерації. Розрахунок по цьому завданню дозволив досліджувати максимальну ємність стека досліджуваних мов: розрахунок ack (1,1, n-1) вимагає n-й вкладеності викликів і дуже швидкий. У Таблиці 2 представлені результати розрахунку пентацен ack (5,2,3), для тих мов, стек яких зміг його (вкладеність викликів 65539) витримати.

Таблиця 2

gcc -O5	1
icc -fast	2.18
clang -O3	2.76
gcc -O0	7.75
icc -O0	8.36
clang -O0	9.64
Erlang	18.51
ghc -O	50.18
lua	122.55
php	423.64

За результатами дослідження зроблено такі висновки, деякі з яких виявилися дещо несподіваними:

- Швидкість компіляції та виконання програм на javascript в популярних браузерах лише в 2-3 рази поступається кращим трансляторам і перевершує навіть деякі якісні компілятори, безумовно набагато(більш ніж в 10 разів) випереджаючи більшість трансляторів інших мов сценаріїв і подібних до них за швидкістю виконання програм;
- Швидкість кодів, що генеруються компілятором мови C фірми Intel, виявилася помітно меншою, ніж компілятор GNU і іноді LLVM;
- Оптимізація кодів краще працює на процесорі Intel;
- Швидкість виконання на процесорі Intel була майже завжди вище, за винятком мови Lisp, Erlang.
- Стек більшості тестованих мов, підтримують тільки дуже обмежене число рекурсивних викликів. Деякі транслятори дозволяють збільшити розмір стека зміною змінних середовища виконання або параметром.

ПЕРЕЛІК ПОСИЛАНЬ:

1. https://visualgo.net/en
2. https://github.com/liuxinyu95/AlgoXY
3.